EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

Monte Carlo Vortical Smoothed Particle Hydrodynamics for Simulating Turbulent Flows

Xingyu Ye[1,2], Xiaokun Wang[1,2,†], Yanrui Xu[1,3], Jiří Kosinka[3], Alexandru C. Telea[4], Lihua You[2], Jian Jun Zhang[2], Jian Chang[2,†]

[1]School of Intelligence Science and Technology, University of Science and Technology Beijing, China
[2]National Centre for Computer Animation, Bournemouth University, United Kingdom
[3]Bernoulli Institute, University of Groningen, Netherlands
[4]Department of Information and Computing Sciences, Utrecht University, Netherlands

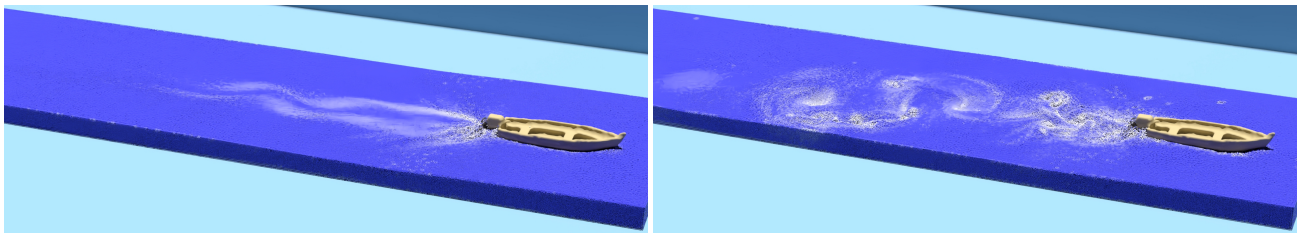


Figure 1: *The divergence-free SPH (DFSPH) method (left) produces basic wake flow motions. Our MCVSPH method (right) effectively generates intricate vortical motions throughout the fluid domain, exhibiting transportation, merging, and splitting of vortices.*

Abstract

For vortex particle methods relying on SPH-based simulations, the direct approach of iterating all fluid particles to capture velocity from vorticity can lead to a significant computational overhead during the Biot-Savart summation process. To address this challenge, we present a Monte Carlo vortical smoothed particle hydrodynamics (MCVSPH) method for efficiently simulating turbulent flows within an SPH framework. Our approach harnesses a Monte Carlo estimator and operates exclusively within a pre-sampled particle subset, thus eliminating the need for costly global iterations over all fluid particles. Our algorithm is decoupled from various projection loops which enforce incompressibility, independently handles the recovery of turbulent details, and seamlessly integrates with state-of-the-art SPH-based incompressibility solvers. Our approach rectifies the velocity of all fluid particles based on vorticity loss to respect the evolution of vorticity, effectively enforcing vortex motions. We demonstrate, by several experiments, that our MCVSPH method effectively preserves vorticity and creates visually prominent vortical motions.

CCS Concepts

• ***Computing methodologies*** → *Physical simulation;*

1. Introduction

Fluid animation constitutes a captivating area of research within the computer graphics community. Among various fluid simulation techniques, the Navier-Stokes equations [CF88] serve as the cornerstone for modern fluid solvers. In the conventional operator splitting scheme [Bri15], the particle velocity field $\mathbf{u}^n$ at time step n is updated in two steps. First, non-pressure effects (such as gravity and viscous forces) are applied to calculate an intermediate velocity field $\bar{\mathbf{u}}^n$. Next, $\bar{\mathbf{u}}^n$ is projected onto a divergence-free field. Due to the application of non-pressure forces (and self-advection in Eulerian approaches) without considering incompressibility, divergent velocity components are inevitably generated. As Figure 2 shows, the pressure projection step, which aims to eliminate these divergent components from the velocity field, often leads to a loss of turbulent details [ZBG15].

Visually realistic simulations of inviscid or low-viscosity liquids with high Reynolds numbers require capturing fine turbulent details. Vortex methods are used to simulate turbulent flows in many studies, where vorticity is stored and evolved using vortex particles [SRF05; WL93] for general fluid simulation; vortex filaments [AN05; WP10]

† Corresponding authors: wangxiaokun@ustb.edu.cn, jchang@bournemouth.ac.uk

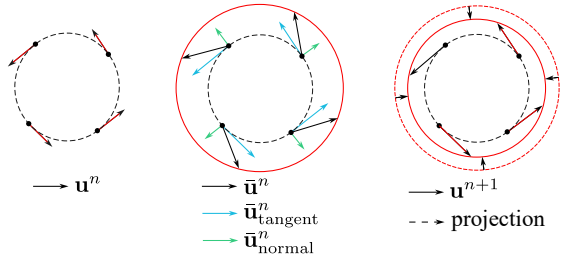


Figure 2: *Left: Velocity field* $\mathbf{u}^n$ *at time step n. Middle: Non-pressure forces push the original velocity* $\mathbf{u}^n$ *into a divergent intermediate velocity* $\bar{\mathbf{u}}^n$*. Right: The pressure force mitigates the divergent (normal) component of* $\bar{\mathbf{u}}^n$*, causing turbulent detail loss.*

for three-dimensional fluid simulation; and vortex sheets [BKB12] for boundaries. While such methods can capture vorticity features and preserve fine turbulent visual details, the computational cost of iterating over the vortex elements is significant [NWRC22]. For instance, the direct approach to integrating the vortex particle method into SPH-based fluid simulation frameworks involves directly considering *all* fluid particles as vortex particles to track vorticity evolution and compute velocity through the Biot-Savart summation. Typical simulations use many fluid particles, so this direct method imposes high computational costs.

Inspired by the work of Rioux-Lavoie et al. [RSÖ*22] using a Monte Carlo estimator to simulate fluids in an Eulerian framework, we propose the Monte Carlo Vortical Smoothed Particle Hydrodynamics (MCVSPH) method to simulate turbulent fluids in an SPH framework. Our method operates independently of pressure projection loops, allowing the use of existing state-of-the-art incompressibility solvers [ICS*14; BK17] and simplifying the process of implementing a complete Lagrangian fluid solver. We incorporate our vortex particle method into an SPH-based framework using the Monte Carlo stochastic method. This adaption allows us to derive a corrective velocity from vorticity using the Biot-Savart summation within a pre-sampled small particle subset, and makes it feasible to implement the vortex particle method efficiently within a pure particle framework, all without the need for expensive global iterations. To organise vortex particles, we treat vortex and SPH particles as separate entities. Following Monte Carlo method principles, we randomly sample a subset of SPH particles and pair each of them with a corresponding vortex particle. This enables us to adopt probabilistic methods to estimate velocities that would otherwise necessitate iterating through all SPH fluid particles.

Our main contributions are summarised as follows:

- A Lagrangian approach enhancing turbulent details;
- Improved Biot-Savart summation in SPH with a Monte Carlo method;
- An organisation scheme for vortex particles for turbulent detail preservation.

The remainder of this paper is organised as follows. We start by reviewing related work (Sec. 2) and also briefly introduce SPH and the vortex particle method (Sec. 3). Next, we introduce our MCVSPH approach (Sec. 4). We illustrate our approach by a series of three-dimensional simulation experiments and comparisons with other SPH-based methods for turbulent flow (Sec. 5). Finally, we conclude the paper and provide directions for future work (Sec. 6).

2. Related work

A substantial body of research exists within the computer graphics community focusing on fluid simulation and SPH methods. We next provide a concise review of relevant work focusing on SPH and techniques for enhancing turbulent details.

SPH-based Incompressible Fluids. Simulating visually authentic fluids often requires incompressibility. Many previous studies focus on achieving incompressible fluid simulations through SPH-based methods. Becker and Teschner [BT07] developed an explicit approach to compute pressure from densities using the Tait equation. Yet, the simulated fluids remain weakly compressible. To overcome this, Solenthaler and Pajarola [SP09] proposed a predictive-corrective scheme (PCISPH), iteratively solving particle pressures from density change using a global stiffness constant. Ihmsen et al. [ICS*14] enhanced the pressure-solving iterations with the implicit incompressible SPH (IISPH) using density deviation computed from velocities and computing stiffness for each particle per time step. However, PCISPH and IISPH solve the Pressure Poisson Equation (PPE) without explicitly enforcing a divergence-free velocity field.

Bender and Koschier [BK17] proposed divergence-free SPH (DFSPH), where position updates correct densities and velocity updates mitigate velocity divergence. In our work, we use DFSPH for its ability to enforce invariant densities and divergence-free velocities. Yet, typical SPH-based incompressible fluid solvers exhibit noticeable numerical damping from coarse discretisations [IOS*14], leading to significant loss of turbulent details and the need for additional approaches to preserve vorticity and turbulent motions. Our MCVSPH method enhances turbulent details for general fluid simulations *independently* of the pressure projection process and can be integrated with most existing incompressibility solvers.

Numerical Dissipation Mitigation. The semi-Lagrangian method is commonly employed to achieve accurate advection. Yet, it can inadvertently blur high-frequency fluid motion information due to the use of low-order accuracy interpolation computations. Eulerian-Lagrangian hybrid methods, such as PIC [Har62] and FLIP [ZB05], mitigate numerical dissipation, yet frequent interpolations during particle-grid transfers can cause momentum loss. Jiang et al. [JSS*15] proposed the affine particle-in-cell (APIC) scheme, which considers angular momentum and shearing deformation during grid-particle transfers. Fu et al. [FGG*17] expand upon the APIC method by generalising the local function on each particle. Qu et al. [QLDJ22] compute weights for particle-grid transfers using the power particle method, which ensures uniform particle distributions and volume preservation.

In the realm of pure Eulerian simulation, Qu et al. [QZG*19] introduce the BiMocq scheme which capitalises on dual mesh characteristics and multi-level mapping to effectively compensate error in a back-and-forth manner. Zehnder et al. [ZNT18] propose an advection-reflection scheme applying an energy-preserving reflection operator halfway through the advection step to reduce the dissi-

pation at the projection step. Nabizadeh et al. [NWRC22] enhance the advection-reflection scheme by incorporating covectors to preserve circulation. Previous efforts to mitigate numerical dissipation primarily address advection dissipation present in Eulerian methods. In contrast, our work focuses on maintaining intricate turbulent details within a Lagrangian framework.

Vorticity Confinement. The vorticity confinement method, introduced to the computer graphics community by Fedkiw et al. [FSJ01], amplifies existing vortices using corrective forces. Letine et al. [LAF11] enhance the computation of these forces for improved energy and momentum conservation. While initially proposed for grid-based scenarios [JKB*10; WM19], vorticity confinement was applied to SPH-based methods fluids by Macklin and Müller [MM13] to enhance particle vortical motions. Although the vorticity confinement method is relatively simple to implement, its sensitivity to parameters can result in the generation of excessive energy and unstable simulations. Moreover, since the amplification forces are contingent on existing vortices, this approach cannot generate new vortices, which restricts its use in simulating fluids with rich turbulent features.

Micropolar Fluids. The micropolar fluid method [Eri66; Luk99] describes fluid micro-structures with non-symmetric stress tensors. This approach finds applications in areas where fluid micro-motion is significant, such as heat transfer [PMZ22], blood flows [KSPS20], and magnetised fluids [ANK22]. Bender et al. [BKKW19] introduced the micropolar method to computer graphics and applied it to SPH-based methods (MPSPH) to simulate turbulent flows. However, as pointed out by Chen et al. [CLL10], micro-rotations represent non-solid-body-like rotation (gyration). Given that this method does not explicitly model solid-body-like rotation (vorticity), the resulting turbulent features often manifest themselves as small-scale rotational motions. Our method instead employs the vorticity equation to explicitly model vortical motions.

Non-Navier-Stokes Methods. To circumvent the challenges faced by general Navier-Stokes solvers, such as vorticity loss, some methods have turned to non-Navier-Stokes methods. Chern et al. [CKP*16] introduced Clebsch variables and the Schrödinger equation to computer graphics. While conventional fluid solvers use velocity or vorticity for direct fluid motion simulation, this method evolves the wave function and extracts velocity and density information from it. Hence, the need for advection is eliminated in a pure Eulerian setting, resulting in significant improvements in addressing numerical dissipation issues. Subsequent research has expanded the use of the Clebsch method. For example, Yang et al. [FLX*22] combine it with the gauge method to effectively capture coherent vortical structures in incompressible flow. Xiong et al. [XWWZ22] integrate the Clebsch method with a level-set approach to simulate free-surface turbulent flow. However, since these methods are developed based on models that diverge from the Navier-Stokes equation, integrating them with existing high-performance Navier-Stokes fluid solvers is challenging. In contrast, our approach seamlessly integrates with existing Navier-Stokes fluid solvers, enabling the simulation of incompressible and turbulent fluids with minimal changes to current fluid simulation methods.

Vortex Methods. Vortex methods [CK00; MM21] leverage Lagrangian elements, such as vortex particles [SRF05; WL93] and vortex filaments [AN05; WP10], to trace and evolve vorticity. Zhu et al. [ZYF10] simulate vortical fluids by a hybrid SPH-FLIP solver with the vorticity equation solved on local grids. Wang et al. [WLB*20] proposed a turbulence refinement method using the Rankine vortex model to recover the energy lost in rotational degrees of freedom. Zhang et al. [ZBG15] propose the IVOCK scheme to restore lost vorticity and expedite the Biot-Savart summation using the fast multipole algorithm [Dar00]. Liu et al. [LWB*21] adopt the vorticity equation and the stream function to reconstruct turbulent details in SPH fluid simulations, where the Biot-Savart summation traverses vortex particles within the support radius. However, the computational overhead of traversing all vortex particles in the Biot-Savart summation is high. Previous studies have introduced solutions to expedite this process within Eulerian frameworks, such as PPPM [ZB14] and the Monte Carlo method [RSÖ*22]. Yet, the application of vortex methods to SPH frameworks remains relatively unexplored.

Given the effectiveness of the vortex particle method in modelling vortex motion, we harness its potential to generate turbulent motion within SPH-based fluids. Our approach leverages the vortex particle method *within an SPH-based framework* using a Monte Carlo estimator, conducting the Biot-Savart summation via a pre-sampled subset of particles.

3. Fundamentals of SPH and the vortex particle method

As we use SPH-based approaches in our work, we next revisit the governing Navier-Stokes equations and general SPH-based numerical computations for various differential operators needed in fluid simulations (Sec. 3.1). We also cover the basics of the vortex particle method and explain the bottleneck of its use in SPH (Sec. 3.2).

The Navier-Stokes equations [CF88] read

$$\begin{aligned} \frac{D\mathbf{u}}{Dt} &= -\frac{1}{\rho}\nabla p + \upsilon\Delta\mathbf{u} + \mathbf{g}, \\ \nabla\cdot\mathbf{u} &= 0, \end{aligned} \tag{1}$$

where $\frac{D\mathbf{u}}{Dt} = \frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}$ is the material derivative; p is the pressure; υ is the kinematic viscosity; and $\mathbf{g}$ models external forces. The equations govern fluid dynamics and form the basis of fluid solvers. However, for computational purposes, it needs to be discretised.

3.1. SPH discretisation

SPH is used as a spatial discretisation method to numerically approximate the differential operators within the governing equations [KBST19]. For general approximation without any differential operators, values on fluid particle i are evaluated by

$$\mathbf{A}_i = \sum_j \frac{m_j}{\rho_j}\mathbf{A}_j W_{ij}, \tag{2}$$

where j denotes a neighbor particle of particle i; m_j and ρ_j are the mass and the density, respectively, of particle j; and $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j, h)$, where $\mathbf{x}$ denote particle positions, is a kernel function with support radius h. In this paper, we use for W the cubic spline

function [KBST19] given by

$$W(\mathbf{x},h)=\sigma_d\begin{cases}6(a^3-a^2)+1 & 0\le a\le\frac{1}{2}\\ 2(1-a)^3 & \frac{1}{2}<a\le 1\\ 0 & \text{otherwise,}\end{cases} \tag{3}$$

where $a=\frac{\|\mathbf{x}\|}{h}$, and σ_d is a kernel normalisation factor varying with the simulation dimension d: $\sigma_d=\frac{4}{3h}$ for $d=1$; $\sigma_d=\frac{40}{7\pi h^2}$ for $d=2$; and $\sigma_d=\frac{8}{\pi h^3}$ for $d=3$, respectively.

Density. Applying Eqn. (2) to the density ρ leads directly to

$$\rho_i=\sum_j m_j W_{ij}. \tag{4}$$

Vorticity. Vorticity ω is defined as the curl $\nabla\times\mathbf{u}$ of the velocity. In the context of discretisation, two SPH formulations exist for representing this curl. The symmetric curl form is given by

$$(\nabla\times\mathbf{A})_i=\rho_i\sum_j m_j\left(\frac{\mathbf{A}_i}{\rho_i^2}+\frac{\mathbf{A}_j}{\rho_j^2}\right)\times\nabla W_{ij}, \tag{5}$$

Alternatively, with $\mathbf{A}_{ji}=\mathbf{A}_j-\mathbf{A}_i$, the difference curl is given by

$$(\nabla\times\mathbf{A})_i=\frac{1}{\rho_i}\sum_j m_j\mathbf{A}_{ji}\times\nabla W_{ij}. \tag{6}$$

As Eqn. (6) exhibits lower error in boundary areas [BKKW19], we use this form for vorticity computation in our method.

The difference form of SPH formulations can also be applied to the discretisation of the gradient operator

$$\nabla\mathbf{A}_i=\sum_j\frac{m_j}{\rho_j}\mathbf{A}_{ji}\otimes\nabla W_{ij}, \tag{7}$$

where $\mathbf{m}\otimes\mathbf{n}=\mathbf{m}\mathbf{n}^T$ denotes the dyadic product.

Viscosity. In scenarios where the simulated fluid is not entirely inviscid, the computations of the Laplacian of velocity $\Delta\mathbf{u}$ and the Laplacian of vorticity $\Delta\omega$ become necessary. We use the following SPH approximation to discretise the Laplacian operator:

$$\Delta\mathbf{A}_i=2(d+2)\sum_j\frac{m_j}{\rho_j}\frac{\mathbf{A}_{ij}\cdot\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2+0.01h^2}\nabla W_{ij}, \tag{8}$$

where $\mathbf{x}_{ij}=\mathbf{x}_i-\mathbf{x}_j$, and d is the spatial dimension of the simulated scenario. This approach reduces the order of the differential equation, enhances numerical stability, and helps conserve linear and angular momenta [Mon92].

Pressure. The pressure force enforces incompressibility of the fluid field. In our work, we opt for the divergence-free SPH solver (DFSPH) [BK17], which maintains constant density and a divergence-free velocity field.

Boundary handling. We address interactions between fluid and surrounding solids by the approach of Akinci et al. [AIA*12]. This involves representing solid boundaries using particles, treating them similarly to fluid particles. The smoothed mass of a solid boundary particle i with density ρ_0 and neighbouring boundary particles k is computed as

$$\Psi_i=\frac{\rho_0 m_i}{\sum_k m_k W_{ik}}=\frac{\rho_0}{\sum_k W_{ik}}, \tag{9}$$

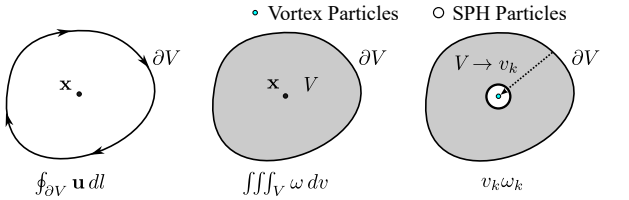


Figure 3: *Left: Circulation loop. The vortex strength vector β is the circulation around position $\mathbf{x}$. Middle: Stokes' theorem converts a surface integral into a volume integral. Right: Convergence to a vortex particle. The loop size is small enough so that volume integration can be approximated by the product of a constant vorticity and the volume.*

where m_i is the non-smoothed mass of particle i.

3.2. The vortex particle method

The vortex particle method has two key components: the vortex strength vector and the Biot-Savart summation, as follows.

The vortex strength vector β represents the circulation at a specific position $\mathbf{x}$ within the fluid field. In fluid dynamics, the general circulation is computed as

$$\beta=\oint_{\partial V}\mathbf{u}\,dl, \tag{10}$$

where ∂V is a closed curve in 2D or a closed surface in 3D surrounding position $\mathbf{x}$ (see Fig. 3 left). Using Stokes' theorem, we can transform Eqn. (10) in 3D to

$$\beta=\iiint_V\nabla\times\mathbf{u}\,dv=\iiint_V\omega\,dv, \tag{11}$$

where V is the volume enclosed by the surface ∂V (see Fig. 3 middle). Ideally, when computing β at a given position $\mathbf{x}$, the closed surface ∂V and the enclosed volume V should be infinitesimal. In practice, V should be small enough so that ω can be considered constant when evaluating β. In this case, the vortex strength vector β of vortex particle k can be approximated as

$$\beta_k\approx\omega_k v_k, \tag{12}$$

where ω_k is the vorticity of vortex particle k and v_k is its (small) volume, respectively (see Fig. 3 right).

The Biot-Savart summation [WL93] recovers the velocity field from vorticity as

$$\mathbf{u}=(\mathbf{K}\times)*\tilde{\omega}, \tag{13}$$

where $\mathbf{K}\times$ represents the Biot-Savart kernel, $*$ denotes the convolution operator, and $\tilde{\omega}$ is the approximated vorticity. In three-dimensional scenarios, the Biot-Savart kernel is given by

$$\mathbf{K}\times=\mathbf{K}(\mathbf{x})\times=-\frac{\mathbf{x}}{4\pi\|\mathbf{x}\|^3}\times. \tag{14}$$

Expanding the convolution operator $*$, Eqn. (13) can be written as

$$\mathbf{u}(\mathbf{x})=\iiint_\chi\mathbf{K}(\mathbf{x}-\mathbf{x}')\times\tilde{\omega}(\mathbf{x}')\,d\mathbf{x}', \tag{15}$$

where χ denotes the domain enclosed by ∂V.

However, Eqn. (15) involves an integration that is not conducive to numerical computation. To discretise this equation, we use the vortex particle method, wherein vortices are abstracted as particles that store position and vortex strength information. The approximated vorticity $\tilde{\omega}$ is represented as

$$\tilde{\omega}(\mathbf{x}) = \sum_{k=1}^{N_p} \beta_k \delta(\mathbf{x} - \mathbf{x}_k), \tag{16}$$

where $\mathbf{x}_k$ denotes the position of vortex particle k; δ is the Dirac delta function; and N_p is the total number of vortex particles used to cover the whole fluid domain.

By substituting Eqn. (16) into the Biot-Savart law (Eqn. (15)), the velocity computation becomes

$$\begin{aligned} \mathbf{u}(\mathbf{x}) &= \iiint_{\chi} \mathbf{K}(\mathbf{x} - \mathbf{x}') \times \left(\sum_{k=1}^{N_p} \beta_k \delta(\mathbf{x}' - \mathbf{x}_k) \right) d\mathbf{x}' \\ &= \sum_{k=1}^{N_p} \mathbf{K}(\mathbf{x} - \mathbf{x}_k) \times \beta_k = \sum_{k=1}^{N_p} -\frac{\mathbf{x} - \mathbf{x}_k}{4\pi \|\mathbf{x} - \mathbf{x}_k\|^3} \times \beta_k. \end{aligned} \tag{17}$$

This shows that the velocity field is recovered from vorticity by iterating over all vortex particles. In SPH, physical attributes (such as position, velocity, and pressure) of a fluid are tracked using a collection of SPH fluid particles – which could also serve as vortex particles that hold vorticity information. Yet, due to the high number of fluid particles, iterating over all these particles becomes impractical when applying the vortex particle method to SPH. An optimised approach, which we present next, solves this issue.

4. MCVSPH Scheme

We apply the vortex particle method in SPH-based fluid simulations to recover turbulent features. For this, we use the vorticity equation for vorticity evolution to compute vorticity loss in the velocity field (Sec. 4.1). We optimise computing velocity corrections from vorticity loss by Monte Carlo sampling, as described next (Sec. 4.2).

4.1. Vorticity loss

To recover turbulent details, we start from the vorticity equation representing the time evolution of vorticity $\omega = \nabla \times \mathbf{u}$

$$\frac{D\omega}{Dt} = (\omega \cdot \nabla)\mathbf{u} + \upsilon \Delta \omega, \tag{18}$$

where υ is the same kinematic viscosity as the one in Eqn. (1). The first term $(\omega \cdot \nabla)\mathbf{u}$ gives the vortex stretching governing the evolution of vortex filaments.

In an ideal scenario, the curl $\nabla \times \mathbf{u}$ of velocity is identical to the vorticity ω. This is not the case in practice, due to distinct equations governing the evolution of velocity and vorticity and the numerical dissipation during the splitting scheme for updating velocity. Zhang [ZBG15] proposed a method that evaluates lost vorticity by the difference between updated vorticity and the curl of updated velocity. Inspired by this, in our SPH-based approach, we 'save' the advection step and approximate differential operators by SPH formulations to evaluate vorticity loss. Specifically, we quantify vorticity loss for each particle by the following steps:

- Compute vorticity ω^n at current time step n using the curl of current velocity field $\omega^n = \nabla \times \mathbf{u}^n$;
- Evolve current vorticity ω^n with the vorticity equation (18) to obtain the vorticity ω^{n+1} at the next time step;
- Apply non-pressure forces $\mathbf{f}_{\text{non-pressure}}$ to the current velocity $\mathbf{u}^n$ to acquire the intermediate velocity $\bar{\mathbf{u}}^n$ by

$$\bar{\mathbf{u}}^n = \mathbf{u}^n + \Delta t \, \mathbf{f}_{\text{non-pressure}}; \tag{19}$$

- Quantify the vorticity loss for each fluid particle as

$$\omega_{\text{loss}}^{n+1} = \omega^{n+1} - \nabla \times \bar{\mathbf{u}}^n. \tag{20}$$

It is worth noting that we have the flexibility to choose either $\nabla \times \bar{\mathbf{u}}^n$ or $\nabla \times \mathbf{u}^{n+1}$ as the second term in Eqn. (20), where $\mathbf{u}^{n+1}$ is the pressure projection outcome of $\bar{\mathbf{u}}^n$, since the curl of the pressure gradient is zero. We adopt the difference curl (Eqn. (6)) to discretise the velocity curl as

$$(\nabla \times \mathbf{u})_i = \frac{1}{\rho_i} \sum_j m_j (\mathbf{u}_j - \mathbf{u}_i) \times \nabla W_{ij}. \tag{21}$$

Computing the vortex-stretching term needs the gradient of velocity $\nabla \mathbf{u}$, which is given by

$$\nabla \mathbf{u}_i = \sum_j \frac{m_j}{\rho_j} (\mathbf{u}_j - \mathbf{u}_i) \otimes \nabla W_{ij}. \tag{22}$$

For viscid fluids, we compute viscosity with the Laplacian of velocity and vorticity using Eqn. (8). Putting it all together, Alg. 1 summarises the computation of vorticity loss.

Algorithm 1 Vorticity loss evaluation

Input: $\Delta t, \mathbf{u}^n$ **Output:** $\omega_{\text{loss}}^{n+1}$

1: **for all** SPH fluid particles i **do**
2: $\omega_i^n \leftarrow (\nabla \times \mathbf{u})_i^n$ ▷ Eqn. (21)
3: $\bar{\mathbf{u}}_i^n \leftarrow \mathbf{u}_i^n + \Delta t \mathbf{f}_{\text{non-pressure}}$ ▷ Eqn. (19)
4: **for all** SPH fluid particles i **do**
5: $\bar{\omega}_i^n \leftarrow \text{VortexStretch}(\Delta t, \omega_i^n, \bar{\mathbf{u}}_i^n)$ ▷ Eqn. (22)
6: $\omega_i^{n+1} \leftarrow \text{Viscosity}(\bar{\omega}_i^n)$ ▷ Eqn. (8) if viscid
7: $\omega_{\text{loss,i}}^{n+1} = \omega_i^{n+1} - \nabla \times \bar{\mathbf{u}}_i^n$ ▷ Eqn. (20)

4.2. Vortex particles with Monte Carlo estimators in SPH

The Monte Carlo method is an effective technique for randomly sampling a field – velocity $\mathbf{u}$, in our case – based on a stochastic distribution and estimating the required value. Rioux-Lavoie et al. [RSÖ*22] improve the discretised Biot-Savart law (Eqn. (17)) with Monte Carlo estimation in an Eulerian framework via

$$\mathbf{u}(t, \mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{K}(\mathbf{x} - \mathbf{y}_i) \times \beta_i}{P(\mathbf{y}_i | t, \mathbf{x})}, \tag{23}$$

where $\mathbf{y}_i$ are all sampled N positions measured in the above-mentioned Eulerian grid – that is, much fewer than the total number of globally distributed vortex particles N_p used in Eqn. (16); and P denotes the probability of the corresponding position $\mathbf{x}$.

We adapt the Monte Carlo estimator to particle methods, making it feasible to correct the velocity field with the vortex particle method. In our method, rather than iterating over all N_f SPH fluid particles,

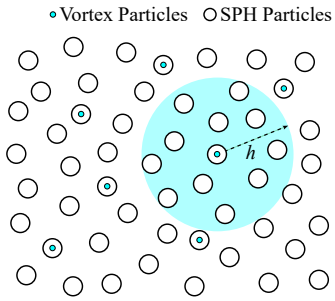


Figure 4: *Vortex particle organisation. Every sampled SPH fluid particle (white, with volume V_k) is paired with a vortex particle (cyan, with volume cV_k) as a cohesive moving unit. The smoothed vorticity loss $\hat{\omega}_{\text{loss}}$ of the vortex particle is computed using SPH (28).*

we stochastically sample $N \ll N_f$ fluid particles to form a small subset Γ of the entire particle-set. Each sampled SPH fluid particle is paired with a single vortex particle as a moving unit (Fig. 4). Since implementing an infinitely small vortex particle is numerically impractical, we assign a small fixed volume v_k to each vortex particle k in Γ, given by

$$v_k = cV_k, \tag{24}$$

where V_k is the volume of the corresponding SPH fluid particle, and $c \in (0,1)$, which we next call the *volume coefficient*, controls the volume of the vortex particle.

By substituting Eqns. (12) and (24) into the Monte Carlo-sampled Biot-Savart formulation (Eqn. (23)), we obtain

$$\mathbf{u}_i = \frac{cV_k}{N} \sum_{k=1}^{N} \frac{\mathbf{K}(\mathbf{x}_i - \mathbf{x}_k) \times \omega_k}{P_k}. \tag{25}$$

Similarly, by converting the vorticity ω_k to the smoothed vorticity loss $\hat{\omega}_{\text{loss},k}$, we derive the velocity correction $\mathbf{u}_{\text{corr,i}}$ for SPH particle i as

$$\mathbf{u}_{\text{corr},i} = \frac{cV_k}{N} \sum_{k=1}^{N} \frac{\mathbf{K}(\mathbf{x}_i - \mathbf{x}_k) \times \hat{\omega}_{\text{loss},k}}{P_k}, \tag{26}$$

where N is the number of vortex particles and P_k is the probability of vortex particle k, given by a stochastic distribution. In practice, we use the uniform distribution

$$P_k = \frac{1}{N_f}, \tag{27}$$

where N_f is the number of fluid particles from which Γ is sampled.

By replacing $\mathbf{A}$ in Eqn. (2) with the vorticity loss, we find the smoothed vorticity loss $\hat{\omega}_{\text{loss,k}}$ of vortex particle k as

$$\hat{\omega}_{\text{loss},k} = \sum_j \frac{m_j}{\rho_j} \omega_{\text{loss},j} W_{kj}, \tag{28}$$

where j are particle k's neighbouring SPH fluid particles (see Fig. 4). For solid particles (used to model boundaries, see Sec. 3.1), we set the vorticity loss to 0.

Algorithm 2 MCVSPH

Input: Δt, $\mathbf{u}^n$, N **Output:** $\mathbf{u}^{n+1}$

1: Subset $\Gamma \leftarrow$ **Sample** and **link** particle pairs ▷ Sec. 4.2
2: **repeat**
3: **for all** SPH fluid particle i **do**
4: $\bar{\mathbf{u}}_i^n \leftarrow \text{ApplyNonPressureForces}(\Delta t, \mathbf{u}^n)$ ▷ Eqn. (19)
5: **for all** SPH fluid particle i **do**
6: $\mathbf{u}_i^{n+1} \leftarrow \text{EnforceIncompressibility}(\Delta t, \bar{\mathbf{u}}^n)$ ▷ DFSPH
7: **for all** SPH fluid particle i **do**
8: $\omega_{\text{loss},i}^{n+1} \leftarrow \text{ComputeVorticityLoss}(\Delta t, \mathbf{u}^n)$ ▷ Alg. 1
9: **for all** vortex particle k in Γ **do**
10: $\hat{\omega}_{\text{loss},k}^{n+1} \leftarrow \text{SmoothVorticityLoss}(\omega_{\text{loss}}^{n+1})$ ▷ Eqn. (28)
11: **for all** SPH fluid particle i **do**
12: $\mathbf{u}_{\text{corr},i}^{n+1} \leftarrow \text{MonteCarloBiotSavart}(\hat{\omega}_{\text{loss}}^{n+1})$ ▷ Eqn. (26)
13: $\mathbf{u}_i^{n+1} \leftarrow \mathbf{u}_i^{n+1} + \mathbf{u}_{\text{corr},i}^{n+1}$ ▷ Eqn. (29)

We leverage the velocity correction $\mathbf{u}_{\text{corr}}$ to compensate the original fluid velocity field and enhance turbulent details by setting

$$\mathbf{u}_i \leftarrow \mathbf{u}_i + \mathbf{u}_{\text{corr},i}. \tag{29}$$

Algorithm 2 outlines our MCVSPH approach. We start by stochastically sampling the fluid particles to yield the subset Γ (line 1). Each sampled fluid particle is linked to a vortex particle; these paired particles next move together as cohesive units. During the simulation loop, we perform five steps, making our approach work independently of pressure-projection loops: First, after applying non-pressure forces (line 4), we enforce incompressibility by methods such as DFSPH [BK17] (line 6). Next, we compute vorticity loss for each fluid particle using the vorticity equation on the current velocity field (line 8). Smoothed vorticity loss values for each vortex particle are computed by an SPH approximation (line 10). Corrective velocities for each fluid particle are computed using the Biot-Savart summation method, enhanced by the Monte Carlo technique (line 12). Finally, these corrective velocities are added to the original velocities to obtain the updated velocity field (line 13).

5. Results

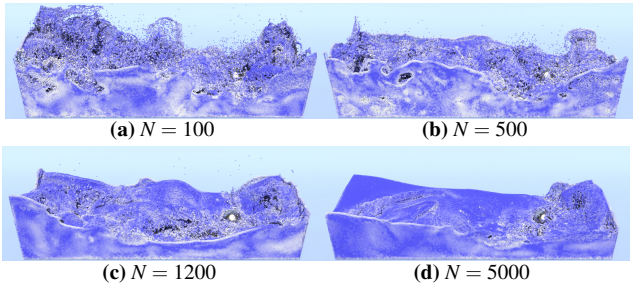
We next present several experiments to validate MCVSPH, which we implemented using the *Taichi* graphics programming language [HLA*19]. All computations are done on a desktop PC with a 3GHz 8-Core Intel Core i7-9700 processor, an NVIDIA Quadro RTX 4000 GPU, and 32GB memory. Rendering uses the *Karma* renderer of Houdini 19.5. All experiments are visualised in the accompanying video.

5.1. Impact of parameters

5.1.1. Effect of sample size N

We next present experiments to evaluate the impact of the sample size N (Sec. 5.1.1) and the volume coefficient c (Sec. 5.1.2) on the performance of turbulence generation of our method.

We evaluate the impact of different sample sizes N in our MCVSPH method by simulations of a dam break scenario with 1.2



(a) $N = 100$ **(b)** $N = 500$ **(c)** $N = 1200$ **(d)** $N = 5000$

Figure 5: *Dam breaking scenario featuring* 1.2 *M SPH fluid particles and a rotating propeller, with particles colour-coded by vorticity (white for high, blue for low vorticity). The volume coefficient c is set to* 0.05*. Results are shown for various sample sizes N:* 100 *(a),* 500 *(b),* 1200 *(c), and* 5000 *(d). Results for* $N = 100$ *and* $N = 500$ *exhibit pronounced noise. In contrast, results for* $N = 1200$ *and* $N = 5000$ *show reduced noise levels and enhanced realism.*

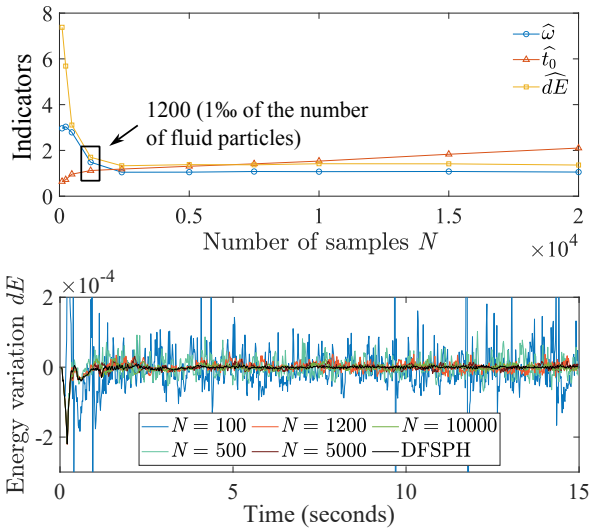
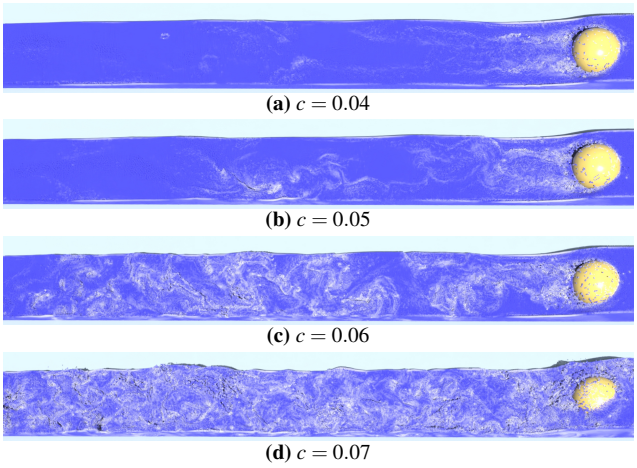


Figure 6: *Indicators evolving with the sample size N for the scenario in Fig. 5. Top: Progression of non-dimensional variables dimensionless vorticity* $\widehat{\omega}$*, time overhead per frame* $\widehat{t_0}$*, and energy variation* $\widehat{dE}$*, for different N values. In all cases, the curves exhibit nonlinear and rapid evolution for* $N < 1200$*. Bottom: Comparison of energy variation evolution for a single particle over time across different sample sizes N. The noise in the energy flattens out as N increases.*

M fluid particles and a rotating propeller at 120 r/min (see Fig. 5). In these experiments, we set the volume coefficient $c = 0.05$ and vary $N \in \{100, 500, 1200, 5000\}$; note that $N = 1200$ corresponds to 1‰ of fluid particles. We visualise the results by rendering particles colour-coded by vorticity (white for high vorticity, blue for low vorticity, see Fig. 5). We see that the fluid simulation becomes noisy and unrealistic for smaller sample sizes ($N = 100$ and $N = 500$). A visually plausible and moderately turbulent flow is obtained for $N = 1200$. Increasing the sample size to $N = 5000$ results in smoothing of turbulent details due to excessive vortex particles with lower vorticity being distributed throughout the fluid field.



(a) $c = 0.04$ **(b)** $c = 0.05$ **(c)** $c = 0.06$ **(d)** $c = 0.07$

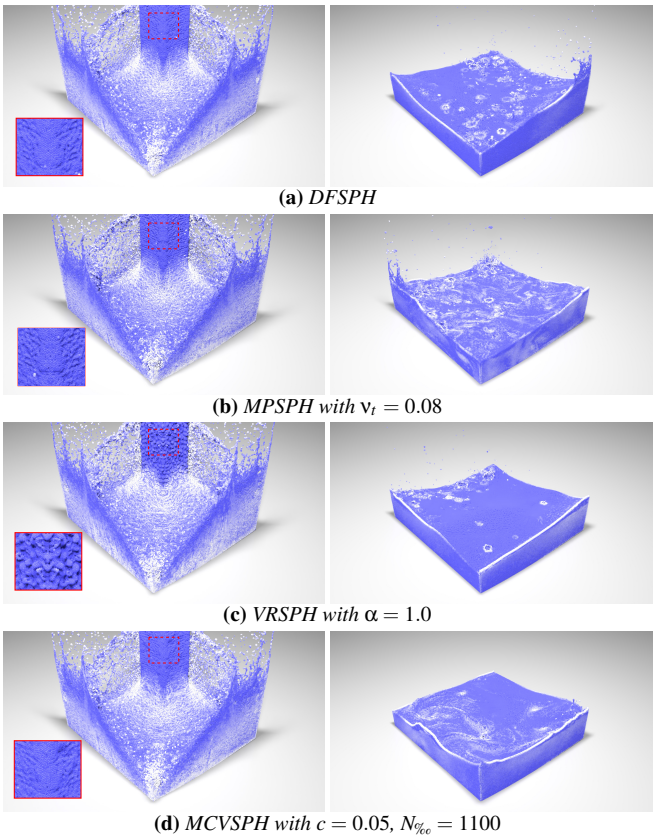
Figure 7: *Illustration of a fast streaming water scenario with* 2.3 *M fluid particles. The simulation employs MCVSPH with a constant sample size* $N_{‰} = 2300$ *and varying volume coefficients* $c \in \{0.04, 0.05, 0.06, 0.07\}$*. As c increases, turbulent details become more pronounced.*

To further assess the effect of the sample size N, and compare our method with DFSPH, we show in Fig. 6(top) the dimensionless vorticity $\widehat{\omega} = \|\omega\|/\|\omega^{\text{DFSPH}}\|$, the dimensionless time overhead per frame $\widehat{t_0} = t_0/t_0^{\text{DFSPH}}$, and the dimensionless energy variation $\widehat{dE} = dE/dE^{\text{DFSPH}}$. Smaller sample sizes (below 1200) lead to significantly higher vorticity values $\widehat{\omega}$; larger sample sizes converge to stable values. Regarding $\widehat{dE}$, smaller sample sizes result in a much noisier and unstable fluid energy variation. Figure 6(bottom) further refines this observation by illustrating the energy variation dE evolving with time for different sample sizes. Analysing $\widehat{t_0}$ (Fig. 6 top), we see a non-linear behaviour when $N < 1200$, with linear evolution for $N > 1200$. Interestingly, the $\widehat{t_0}$ curve for $N < 1200$ says that the time overhead of MCVSPH is even lower than that of DFSPH. This is due to the sparser distribution of fluid particles with smaller sample sizes, leading to fewer density correction iterations in DFSPH. Once the sample size exceeds 1200 and the noise is suppressed, the number of density correction iterations aligns with typical DFSPH, resulting in linear increase in the $\widehat{t_0}$ curve with respect to N. This also demonstrates the impracticability of treating *all* fluid particles as vortex particles due to the heavy computational burden (approximately 63 times slower than DFSPH per frame). By exclusively iterating over a pre-sampled subset $N \ll N_f$ of particles, we substantially reduce the computational overhead, making our method comparable with the baseline DFSPH.

From these results, we see that setting the sample size N to roughly 1‰ of the total fluid particle count yields a well-optimised fluid animation with visually realistic vorticity, reduced noise, and reasonable computation time. We use this setting of N for all further results in this paper, and indicate it next by the notation $N_{‰}$.

5.1.2. Effect of volume coefficient c

To assess the effect of c, we show in Fig. 7 a dynamic scenario featuring fast streaming water with 2.3 M fluid particles, a stationary



(a) *DFSPH*

(b) *MPSPH with* $\nu_t = 0.08$

(c) *VRSPH with* $\alpha = 1.0$

(d) *MCVSPH with* $c = 0.05$, $N_{\text{‰}} = 1100$

Figure 8: *Falling water column with* 1.1 *M fluid particles coloured by vorticity. Comparison of DFSPH, MPSPH with* $\nu_t = 0.08$*, VRSPH with* $\alpha = 1.0$*, and our MCVSPH with volume coefficient* $c = 0.05$ *and sample size* $N_{\text{‰}} = 1100$*. We see that VRSPH creates unrealistic local aggregation patterns resulting from its Biot-Savart summation within local neighbourhoods. Turbulent details get lost quickly in DFSPH and VRSPH. In contrast, MPSPH generates fine ripples on the fluid, and our MCVSPH preserves vortices.*

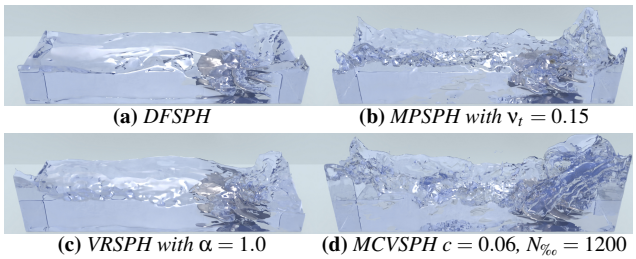
ball, and open boundaries on the left and right sides. Fluid particles are initialised with leftward velocities of 3 m/s. We show results for different volume coefficients $c \in \{0.04, 0.05, 0.06, 0.07\}$ for a constant sample size $N_{\text{‰}} = 2300$.

We see that, as the volume coefficient c increases, the water flow becomes progressively more turbulent – simply put, this shows how one can control the intensity of turbulent flow via tuning c.

5.2. Validation of generating turbulent flows

We further confirm the effectiveness of our MCVSPH approach in capturing turbulent details by five simulation scenarios. We also compare our method with the micropolar SPH (MPSPH) [BKKW19], the vorticity refinement SPH (VRSPH) [LWB*21], and the divergence-free SPH (DFSPH) [BK17] methods.

Falling water column. Our first experiment simulates a falling water column with 1.1 M fluid particles (Fig. 8). The fluid, initially shaped as a box column with a height of 6.0 m and a cross-section



(a) *DFSPH* **(b)** *MPSPH with* $\nu_t = 0.15$

(c) *VRSPH with* $\alpha = 1.0$ **(d)** *MCVSPH* $c = 0.06$, $N_{\text{‰}} = 1200$

Figure 9: *Dam breaking with* 1.2 *M SPH fluid particles and a rotating propeller at* 120 *r/min. Comparison of DFSPH, MPSPH with* $\nu_t = 0.15$*, VRSPH with* $\alpha = 1.0$*, and our MCVSPH with* $c = 0.06$ *and* $N_{\text{‰}} = 1200$*. MCVSPH yields more visible turbulent motions than the other three methods.*

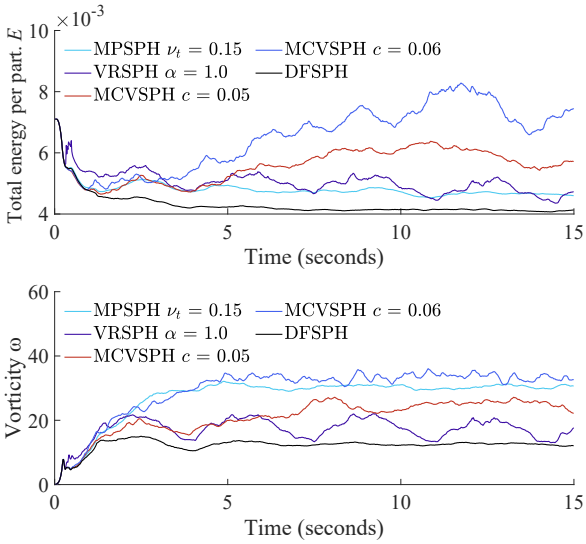
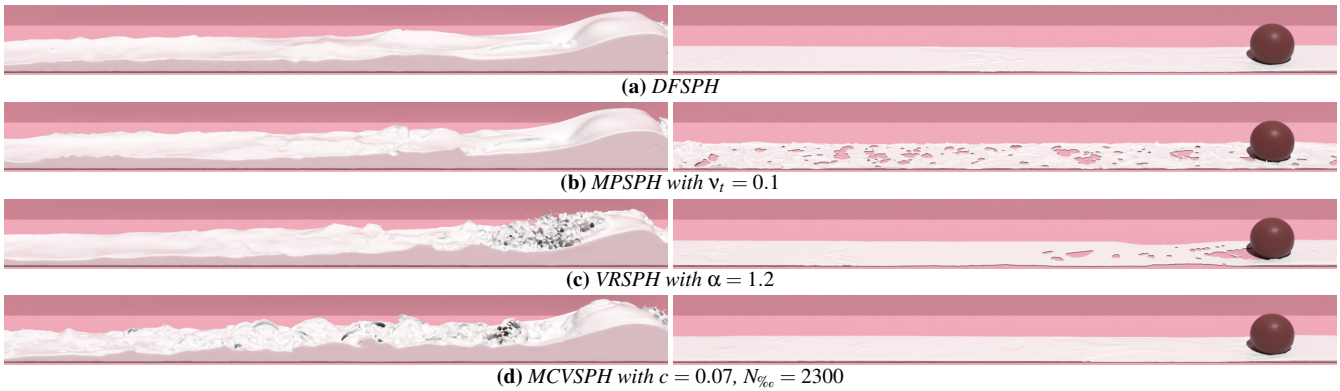


Figure 10: *Comparison of total energy E (top) and vorticity* ω *(bottom) among various methods simulating a propeller scenario (see Fig. 9). MCVSPH produces more total energy for vortical motions with similar vorticity levels compared to other methods (see MCVSPH with* $c = 0.06$ vs *MPSPH with* $\nu_t = 0.15$*).*

area of 4 m^2, experiences free fall under gravity. We compare our MCVSPH approach ($c = 0.05$, $N_{\text{‰}} = 1100$) with DFSPH, MPSPH (with the transfer coefficient $\nu_t = 0.08$), and VRSPH (with the gain $\alpha = 1.0$). From Fig. 8, we see that both DFSPH and VRSPH fail to capture turbulent details effectively, leading to a quick loss of such features. Also, the left image for VRSPH shows noticeable artefacts in the water column, where fluid particles clump together unnaturally in local regions. We attribute this behaviour to VRSPH's usage of the Biot-Savart summation within the support radius of each SPH particle, which causes significant local movements. We also see that both MPSPH and MCVSPH can effectively preserve turbulent details, but with differing patterns: MPSPH creates small-scale ripples, whereas MCVSPH produces visible vortical motions resulting from our vorticity-explicit method.

Dam breaking with a rotating propeller. We further show the effectiveness of MCVSPH by a dynamic scenario involving the



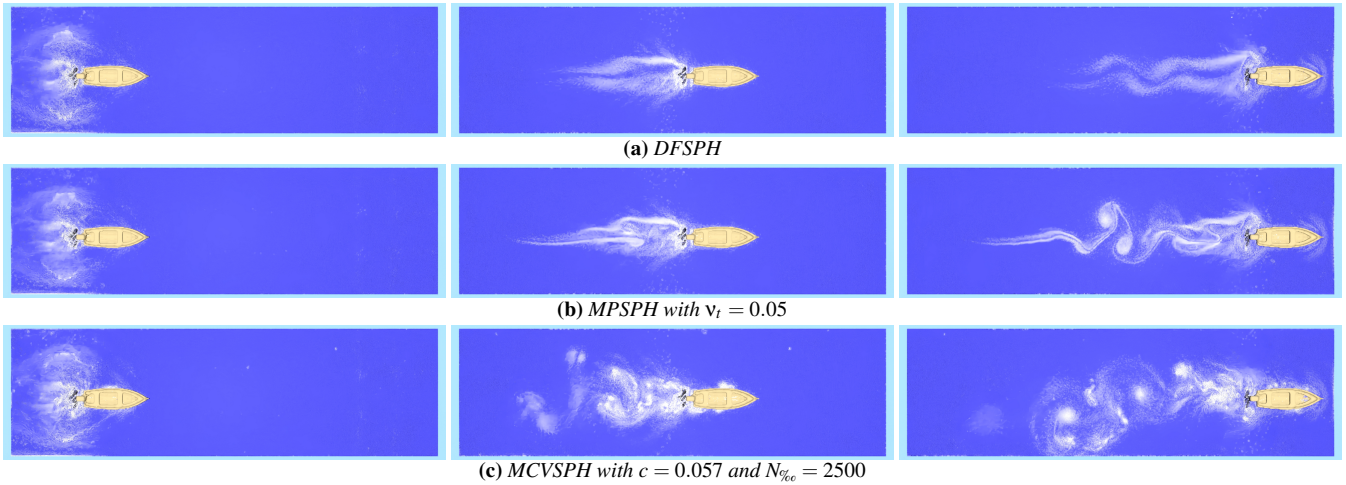
(a) *DFSPH*

(b) *MPSPH with* $\nu_t = 0.1$

(c) *VRSPH with* $\alpha = 1.2$

(d) *MCVSPH with* $c = 0.07$, $N_{\%o} = 2300$

Figure 11: *Fast streaming milk with* 2.3 *M fluid particles and a static chocolate ball (same scenario settings and layout as Fig. 7). Comparison of DFSPH, MPSPH with* $\nu_t = 0.1$*, VRSPH with* $\alpha = 1.2$*, and our MCVSPH with* $c = 0.07$ *and* $N_{\%o} = 2300$*. On the left, our MCVSPH approach generates prominent vortical details on fluid surfaces. As the fluid slows down and becomes shallower, as shown on the right, our MCVSPH yields a realistic simulation of tranquil shallow liquid similar to DFSPH.*



(a) *DFSPH*

(b) *MPSPH with* $\nu_t = 0.05$

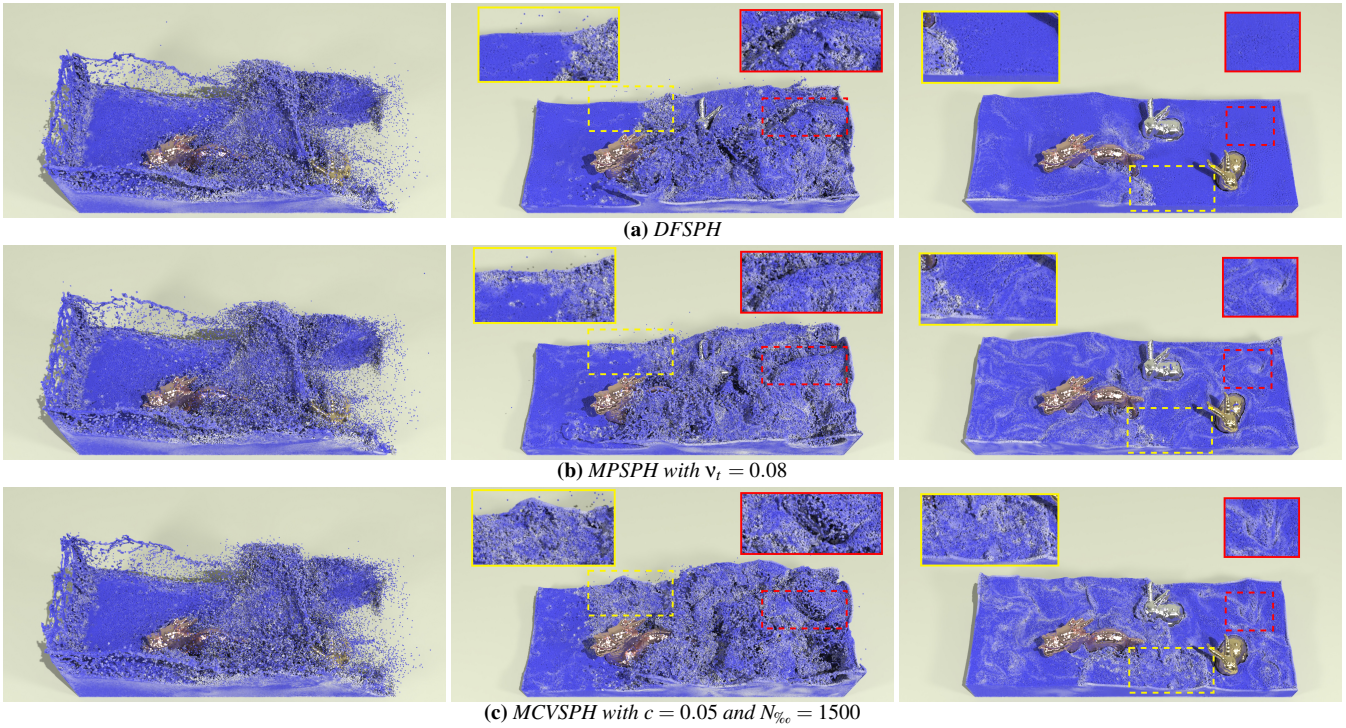
(c) *MCVSPH with* $c = 0.057$ *and* $N_{\%o} = 2500$

Figure 12: *A small boat moving forward with* 2.5 *M vorticity colour-coded fluid particles and a propeller at* 60 *r/min. Comparison of DFSPH, MPSPH with* $\nu_t = 0.05$*, and our MCVSPH with* $c = 0.057$ *and* $N_{\%o} = 2500$*. DFSPH shows basic wake flow from the boat rear. MPSPH yields a more detailed flow. Our MCVSPH yields explicit visible transportation, merging, and splitting of vortices.*

acceleration of 1.2 M fluid particles by a rotating propeller (Fig. 9). The propeller, rotating at 120 r/min, induces turbulent motion in the fluid. We compare our method ($c = 0.06$, $N_{\%o} = 1200$) with DFSPH, MPSPH with $\nu_t = 0.15$, and VRSPH with $\alpha = 1.0$. The results show that VRSPH exhibits fewer turbulent details compared to MPSPH and MCVSPH. MPSPH generates fine turbulent motions, visualised as ripples. Our MCVSPH produces more prominent vortical motions than the other studied methods (see the accompanying video).

The graphs in Figure 10 illustrate the temporal progression of the total energy E per particle (top) and vorticity ω (bottom) over a 15 second period. The bottom graph shows that all compared methods generate a higher vorticity than the baseline DFSPH method. Notably, both our MCVSPH approach with $c = 0.06$ and the MPSPH method with $\nu_t = 0.15$ exhibit similarly elevated vorticity levels, the highest in this comparison. However, the top graph reveals that MCVSPH yields a greater total energy, suggesting that the increased vorticity is primarily attributed to intensified vortical motions in the MCVSPH method. At a higher level, we see that our method does not score significantly negatively with respect to estimation of E or ω with respect to the compared methods. The E and ω differences are small and, as the other images in this paper show, our method produces more visually-realistic effects than the compared methods.

Fast streaming milk with a static chocolate ball. In this simulation (Fig. 11), the setup is identical to the experiment in Fig. 7. We compare DFSPH, MPSPH with $\nu_t = 0.1$, VRSPH with $\alpha = 1.2$, and our MCVSPH with $c = 0.07$ and $N_{\%o} = 2300$ to demonstrate the effectiveness of our method. Observing the left images, we see that both MPSPH and MCVSPH produce rich turbulent details, with MCVSPH showing a more prolonged persistence of turbulent flows compared to MPSPH. As the fluid gradually slows down and becomes shallower (images on the right in Fig. 11), our MCVSPH provides a more credible simulation of a tranquil shallow fluid, more similar to DFSPH as compared with the other studied methods.



(a) *DFSPH*

(b) *MPSPH with* $\nu_t = 0.08$

(c) *MCVSPH with* $c = 0.05$ *and* $N_{‰} = 1500$

Figure 13: *A dam breaking scenario with* 1.5 *M vorticity colour-coded fluid particles, a static Stanford dragon, and two static Stanford bunnies. Comparison of DFSPH, MPSPH with* $\nu_t = 0.08$*, and our MCVSPH with* $c = 0.05$ *and* $N_{‰} = 1500$*. Our MCVSPH approach produces evident vortical movements within the fluid domain, which is visible in the middle and right images.*

Table 1: *Average kinetic energy* E_k*, average total energy* E*, average kinetic energy percentage* $\varepsilon_k = E_k/E \times 100\%$ *per second per particle, and average computational time per frame* t_0 *(three methods, four fluid particle counts* n*, first* 450 *frames). While it is slightly slower than the other two, our approach can produce a larger percentage of kinetic energy, which means our approach has a higher efficiency in generating turbulent motions.*

n [million]	Approach	t_0 [ms]	E_k [$\times 10^{-3}$ s^{-1}]	E	ε_k [%]
1.1 (Fig. 8)	DFSPH	259.9	16.0	47.5	33.7
	MPSPH	276.6	15.1	44.8	33.7
	MCVSPH	295.2	14.3	41.8	34.2
1.2 (Fig. 9)	DFSPH	436.0	2.3	9.9	23.2
	MPSPH	439.7	3.1	10.9	28.4
	MCVSPH	452.4	4.4	14.4	30.6
2.5 (Fig. 12)	DFSPH	459.4	0.103	11.2	0.92
	MPSPH	516.5	0.097	10.0	0.97
	MCVSPH	629.3	0.111	8.2	1.35
1.5 (Fig. 13)	DFSPH	603.8	2.3	8.8	26.1
	MPSPH	625.1	2.3	8.6	26.7
	MCVSPH	629.2	2.7	9.0	30.0

Forward moving boat on tranquil water with a propeller. In this experiment, we simulate a scenario involving a boat moving forward with a propeller at 60 r/min and 2.5 M fluid particles (Fig. 12). We compare three methods: DFSPH, MPSPH with $\nu_t = 0.05$, and our MCVSPH with $c = 0.057$ and $N_{‰} = 2500$. DFSPH generates basic wake flows from the rear of the boat. MPSPH adds more details to the flow. Our MCVSPH approach clearly shows the transportation, merging, and splitting of vortices driven by the propeller at the rear of the boat (see the accompanying video).

Dam breaking with a static Stanford dragon and two static Stanford bunnies. In our final experiment, we simulate a dam breaking scenario with 1.5 M fluid particles alongside a static Stanford dragon and two static Stanford bunnies (Fig. 13). We compare DFSPH, MPSPH with $\nu_t = 0.08$, and our MCVSPH with $c = 0.05$ and $N_{‰} = 1500$. Differences are primarily noticeable in the middle and right figures. The figures illustrate that DFSPH shows almost no turbulent features, while MPSPH and our MCVSPH preserve clear turbulent motions (see accompanying video).

Performance comparison. Table 1 summarises the average computational times t_0 per frame, the average kinetic energy E_k, the average total energy E, and the percentage of kinetic energy ε_k per second per particle of the first 450 frames for the compared methods. Our approach (MCVSPH, bottom row) exhibits a slightly lower speed than MPSPH and DFSPH, as traversals over particle samples are still required. This somewhat limits the scalability of our approach, indicating the need for research into further enhancements. However, the comparison of the percentage of kinetic energy ε_k demonstrates that our approach can generate a higher proportion of kinetic energy per second. Thus, even if it is slightly slower than

the other two methods, our approach can produce turbulent motions with a higher efficiency in general simulation scales.

6. Conclusion

We have presented a Monte Carlo vortical SPH approach for simulating turbulent flows with SPH-based frameworks. Our method works independently of pressure projection loops and flexibly integrates with existing efficient pressure solvers. We leverage the vortex particle method within an SPH-based framework to model turbulent motions. To enhance the efficiency of computing corrective velocity from vorticity loss, we use a Monte Carlo estimator to compute the Biot-Savart summation on a pre-sampled particle subset. We pair vortex particles with a randomly sampled subset of fluid particles making them move together as cohesive units. Vortex particles, which are different in volume from SPH particles, carry smoothed vorticity loss approximated using SPH.

The effectiveness of our method is mainly influenced by the sample size N and the volume c of vortex particles. Setting N to roughly 1‰ of fluid particles yields an optimal balance among turbulent details, computational efficiency, and visual quality. Additionally, increasing the volume c of vortex particles enhances turbulent effects. Our experiments show that our MCVSPH method can effectively generate and preserve visually prominent vortical motions.

While our method enhances turbulent details in SPH-based fluid simulations, several unresolved issues remain. Firstly, in terms of incompressibility and volume preservation, our method does not provide performance gains compared to DFSPH. Additionally, traversals over particle samples are still indispensable in our approach, which limits the scalability to large-scale simulations. Furthermore, as the distribution of particle samples is randomly initialized, this induces slightly stochastic fluid motions in each simulation, even if all parameter settings are the same. Apart from that, a fully robust setting of our method's parameters is still a topic to be explored – again, similar with comparable methods. We aim to develop viable solutions for these challenges in our future research endeavors.

Acknowledgements

We would like to thank the *Taichi* graphics community for the kind support in coding and high-performance computation. This research was supported by National Key Research and Development Program of China (No. 2022ZD0118001), National Natural Science Foundation of China (Nos. 62376025, 62332017, U22A2022), Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515030177), China Scholarship Council, and was partially funded by Horizon 2020-Marie Skłodowska-Curie Action-Individual Fellowships (No. 895941).

References

[AIA*12] AKINCI, NADIR, IHMSEN, MARKUS, AKINCI, GIZEM, et al. "Versatile Rigid-Fluid Coupling for Incompressible SPH". *ACM Transactions on Graphics* 31.4 (July 2012), 1–8. DOI: 10.1145/2185520.2185558 4.

[AN05] ANGELIDIS, ALEXIS and NEYRET, FABRICE. "Simulation of Smoke Based on Vortex Filament Primitives". *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Los Angeles California: ACM, July 2005, 87–96. DOI: 10.1145/1073368.1073380 1, 3.

[ANK22] ABBAS, NADEEM, NADEEM, SOHAIL, and KHAN, MUHAMMAD N. "Numerical Analysis of Unsteady Magnetized Micropolar Fluid Flow over a Curved Surface". *Journal of Thermal Analysis and Calorimetry* 147.11 (June 2022), 6449–6459. DOI: 10.1007/s10973-021-10913-0 3.

[BK17] BENDER, JAN and KOSCHIER, DAN. "Divergence-Free SPH for Incompressible and Viscous Fluids". *IEEE Transactions on Visualization and Computer Graphics* 23.3 (Mar. 2017), 1193–1206. DOI: 10.1109/TVCG.2016.2578335 2, 4, 6, 8.

[BKB12] BROCHU, TYSON, KEELER, TODD, and BRIDSON, ROBERT. "Linear-time Smoke Animation with Vortex Sheet Meshes". SCA '12. Lausanne, Switzerland: Eurographics Association, 2012, 87–95. DOI: 10.5555/2422356.2422371 2.

[BKKW19] BENDER, JAN, KOSCHIER, DAN, KUGELSTADT, TASSILO, and WEILER, MARCEL. "Turbulent Micropolar SPH Fluids with Foam". *IEEE Transactions on Visualization and Computer Graphics* 25.6 (June 2019), 2284–2295. DOI: 10.1109/TVCG.2018.2832080 3, 4, 8.

[Bri15] BRIDSON, ROBERT. *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis, 2015. ISBN: 9781482232837 1.

[BT07] BECKER, MARKUS and TESCHNER, MATTHIAS. "Weakly Compressible SPH for Free Surface Flows". *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '07. San Diego, California: Eurographics Association, 2007, 209–217. DOI: 10.5555/1272690.1272719 2.

[CF88] CONSTANTIN, PETER and FOIAŞ, CIPRIAN. *Navier-Stokes Equations*. Chicago Lectures in Mathematics. Chicago: University of Chicago Press, 1988. ISBN: 978-0-226-11549-8 1, 3.

[CK00] COTTET, GEORGES-HENRI. and KOUMOUTSAKOS, PETROS D. *Vortex Methods: Theory and Practice*. Cambridge ; New York: Cambridge University Press, 2000. ISBN: 978-0-521-62186-1 3.

[CKP*16] CHERN, ALBERT, KNÖPPEL, FELIX, PINKALL, ULRICH, et al. "Schrödinger's Smoke". *ACM Transactions on Graphics* 35.4 (July 2016), 1–13. DOI: 10.1145/2897824.2925868 3.

[CLL10] CHEN, JAMES, LIANG, CHUNLEI, and LEE, JAMES D. "Theory and Simulation of Micropolar Fluid Dynamics". *Proceedings of the Institution of Mechanical Engineers, Part N: Journal of Nanoengineering and Nanosystems* 224.1-2 (Mar. 2010), 31–39. DOI: 10.1177/1740349911400132 3.

[Dar00] DARVE, ERIC. "The Fast Multipole Method: Numerical Implementation". *Journal of Computational Physics* 160.1 (May 2000), 195–240. DOI: 10.1006/jcph.2000.6451 3.

[Eri66] ERINGEN, A. CEMAL. "Theory of Micropolar Fluids". *Journal of Mathematics and Mechanics* 16.1 (1966), 1–18. JSTOR: 24901466 3.

[FGG*17] FU, CHUYUAN, GUO, QI, GAST, THEODORE, et al. "A Polynomial Particle-in-Cell Method". *ACM Transactions on Graphics* 36.6 (Nov. 2017), 222:1–222:12. DOI: 10.1145/3130800.3130878 2.

[FLX*22] FENG, FAN, LIU, JINYUAN, XIONG, SHIYING, et al. "Impulse Fluid Simulation". *IEEE Transactions on Visualization and Computer Graphics* (2022), 1–1. DOI: 10.1109/TVCG.2022.3149466 3.

[FSJ01] FEDKIW, RONALD, STAM, JOS, and JENSEN, HENRIK WANN. "Visual Simulation of Smoke". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, 15–22. DOI: 10.1145/383259.383260 3.

[Har62] HARLOW, FRANCIS H. *The particle-in-cell method for numerical solution of problems in fluid dynamics*. Technical Report, Los Alamost National Labs. Mar. 1962. DOI: 10.2172/4769185 2.

[HLA*19] HU, YUANMING, LI, TZU-MAO, ANDERSON, LUKE, et al. "Taichi: A Language for High-Performance Computation on Spatially Sparse Data Structures". *ACM Transactions on Graphics* 38.6 (Nov. 2019), 201:1–201:16. DOI: 10.1145/3355089.3356506 6.

[ICS*14] IHMSEN, MARKUS, CORNELIS, JENS, SOLENTHALER, BARBARA, et al. "Implicit Incompressible SPH". *IEEE Transactions on Visualization and Computer Graphics* 20.3 (Mar. 2014), 426–435. DOI: 10.1109/TVCG.2013.105 2.

[IOS*14] IHMSEN, MARKUS, ORTHMANN, JENS, SOLENTHALER, BARBARA, et al. "SPH Fluids in Computer Graphics". *Eurographics 2014 - State of the Art Reports*. Ed. by LEFEBVRE, SYLVAIN and SPAGNUOLO, MICHELA. The Eurographics Association, 2014. DOI: 10.2312/egst.20141034 2.

[JKB*10] JANG, TAEKWON, KIM, HEEYOUNG, BAE, JINHYUK, et al. "Multilevel Vorticity Confinement for Water Turbulence Simulation". *The Visual Computer* 26.6-8 (June 2010), 873–881. DOI: 10.1007/s00371-010-0487-1 3.

[JSS*15] JIANG, CHENFANFU, SCHROEDER, CRAIG, SELLE, ANDREW, et al. "The Affine Particle-in-Cell Method". *ACM Transactions on Graphics* 34.4 (July 2015), 51:1–51:10. DOI: 10.1145/2766996 2.

[KBST19] KOSCHIER, DAN, BENDER, JAN, SOLENTHALER, BARBARA, and TESCHNER, MATTHIAS. "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids". *Eurographics 2019 - Tutorials*. Ed. by JAKOB, WENZEL and PUPPO, ENRICO. The Eurographics Association, 2019. DOI: 10.2312/egt.20191035 3, 4.

[KSPS20] KARVELAS, EVANGELOS, SOFIADIS, GIORGOS, PAPATHANASIOU, THANASIS, and SARRIS, IOANNIS. "Effect of Micropolar Fluid Properties on the Blood Flow in a Human Carotid Model". *Fluids* 5.3 (July 2020), 125. DOI: 10.3390/fluids5030125 3.

[LAF11] LENTINE, MICHAEL, AANJANEYA, MRIDUL, and FEDKIW, RONALD. "Mass and Momentum Conservation for Fluid Simulation". *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Vancouver British Columbia Canada: ACM, Aug. 2011, 91–100. DOI: 10.1145/2019406.2019419 3.

[Luk99] LUKASZEWICZ, GRZEGORZ. *Micropolar Fluids: Theory and Applications*. Modeling and Simulation in Science, Engineering and Technology. Boston: Birkhäuser, 1999. ISBN: 978-0-8176-4008-8 3.

[LWB*21] LIU, SINUO, WANG, XIAOKUN, BAN, XIAOJUAN, et al. "Turbulent Details Simulation for SPH Fluids via Vorticity Refinement". *Computer Graphics Forum* 40.1 (2021), 54–67. DOI: 10.1111/cgf.14095 3, 8.

[MM13] MACKLIN, MILES and MÜLLER, MATTHIAS. "Position Based Fluids". *ACM Transactions on Graphics* 32.4 (July 2013), 104:1–104:12. DOI: 10.1145/2461912.2461984 3.

[MM21] MIMEAU, CHLOÉ and MORTAZAVI, IRAJ. "A Review of Vortex Methods and Their Applications: From Creation to Recent Advances". *Fluids* 6.2 (Feb. 2021), 68. DOI: 10.3390/fluids6020068 3.

[Mon92] MONAGHAN, JOSEPH. "Smoothed Particle Hydrodynamics". *Annual Review of Astronomy and Astrophysics* 30.1 (Sept. 1992), 543–574. DOI: 10.1146/annurev.aa.30.090192.002551 4.

[NWRC22] NABIZADEH, MOHAMMAD SINA, WANG, STEPHANIE, RAMAMOORTHI, RAVI, and CHERN, ALBERT. "Covector Fluids". *ACM Transactions on Graphics* 41.4 (July 2022), 113:1–113:16. DOI: 10.1145/3528223.3530120 2, 3.

[PMZ22] PASHA, POOYA, MIRZAEI, SAEID, and ZARINFAR, MEYSAM. "Application of Numerical Methods in Micropolar Fluid Flow and Heat Transfer in Permeable Plates". *Alexandria Engineering Journal* 61.4 (Apr. 2022), 2663–2672. DOI: 10.1016/j.aej.2021.08.040 3.

[QLDJ22] QU, ZIYIN, LI, MINCHEN, DE GOES, FERNANDO, and JIANG, CHENFANFU. "The Power Particle-in-Cell Method". *ACM Transactions on Graphics* 41.4 (July 2022), 118:1–118:13. DOI: 10.1145/3528223.3530066 2.

[QZG*19] QU, ZIYIN, ZHANG, XINXIN, GAO, MING, et al. "Efficient and Conservative Fluids Using Bidirectional Mapping". *ACM Transactions on Graphics* 38.4 (July 2019), 128:1–128:12. DOI: 10.1145/3306346.3322945 2.

[RSÖ*22] RIOUX-LAVOIE, DAMIEN, SUGIMOTO, RYUSUKE, ÖZDEMIR, TÜMAY, et al. "A Monte Carlo Method for Fluid Simulation". *ACM Transactions on Graphics* 41.6 (Nov. 2022), 240:1–240:16. DOI: 10.1145/3550454.3555450 2, 3, 5.

[SP09] SOLENTHALER, BARBARA and PAJAROLA, RENATO. "Predictive-corrective Incompressible SPH". *ACM SIGGRAPH 2009 Papers*. SIGGRAPH '09. New Orleans, Louisiana: Association for Computing Machinery, 2009. DOI: 10.1145/1576246.1531346 2.

[SRF05] SELLE, ANDREW, RASMUSSEN, NICK, and FEDKIW, RONALD. "A Vortex Particle Method for Smoke, Water and Explosions". *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. Los Angeles, California: Association for Computing Machinery, 2005, 910–914. DOI: 10.1145/1186822.1073282 1, 3.

[WL93] WINCKELMANS, GRÉGOIRE S. and LEONARD, ANTHONY. "Contributions to Vortex Particle Methods for the Computation of Three-Dimensional Incompressible Unsteady Flows". *Journal of Computational Physics* 109.2 (Dec. 1993), 247–273. DOI: 10.1006/jcph.1993.1216 1, 3, 4.

[WLB*20] WANG, XIAOKUN, LIU, SINUO, BAN, XIAOJUAN, et al. "Robust Turbulence Simulation for Particle-based Fluids Using the Rankine Vortex Model". *The Visual Computer* 36.10 (2020), 2285–2298. DOI: 10.1007/s00371-020-01914-5 3.

[WM19] WEN, JINGHUAN and MA, HUIMIN. "Real-Time Smoke Simulation Based on Vorticity Preserving Lattice Boltzmann Method". *The Visual Computer* 35.9 (Sept. 2019), 1279–1292. DOI: 10.1007/s00371-018-1514-x 3.

[WP10] WEISSMANN, STEFFEN and PINKALL, ULRICH. "Filament-Based Smoke with Vortex Shedding and Variational Reconnection". *ACM Transactions on Graphics* 29.4 (July 2010), 115:1–115:12. DOI: 10.1145/1778765.1778852 1, 3.

[XWWZ22] XIONG, SHIYING, WANG, ZHECHENG, WANG, MENGDI, and ZHU, BO. "A Clebsch Method for Free-Surface Vortical Flow Simulation". *ACM Transactions on Graphics* 41.4 (July 2022), 116:1–116:13. DOI: 10.1145/3528223.3530150 3.

[ZB05] ZHU, YONGNING and BRIDSON, ROBERT. "Animating Sand as a Fluid". *ACM Transactions on Graphics* 24.3 (July 2005), 965–972. DOI: 10.1145/1073204.1073298 2.

[ZB14] ZHANG, XINXIN and BRIDSON, ROBERT. "A PPPM Fast Summation Method for Fluids and Beyond". *ACM Transactions on Graphics* 33.6 (Nov. 2014), 206:1–206:11. DOI: 10.1145/2661229.2661261 3.

[ZBG15] ZHANG, XINXIN, BRIDSON, ROBERT, and GREIF, CHEN. "Restoring the Missing Vorticity in Advection-Projection Fluid Solvers". *ACM Transactions on Graphics* 34.4 (July 2015), 52:1–52:8. DOI: 10.1145/2766982 1, 3, 5.

[ZNT18] ZEHNDER, JONAS, NARAIN, RAHUL, and THOMASZEWSKI, BERNHARD. "An Advection-Reflection Solver for Detail-Preserving Fluid Simulation". *ACM Transactions on Graphics* 37.4 (July 2018), 85:1–85:8. DOI: 10.1145/3197517.3201324 2.

[ZYF10] ZHU, BO, YANG, XUBO, and FAN, YE. "Creating and Preserving Vortical Details in SPH Fluid". *Computer Graphics Forum* 29.7 (2010), 2207–2214. DOI: 10.1111/j.1467-8659.2010.01809.x 3.